To appear in Journal of Internet Research

ALIBI: A Novel Approach to Resource Discovery

David Flater[1]

and

Yelena Yesha
University of Maryland Baltimore County
Baltimore, MD 21228 U.S.A.
Phone: 410-455-3542
Fax: 410-455-3969
E-mail: yeyesha@cs.umbc.edu

Abstract

The exponential growth of the Internet has resulted in what is known as the resource discovery problem. Although the network makes it possible for users to retrieve enormous amounts of information, it provides insufficient support for locating the specific information that is needed. Various tools have emerged that either simplify the task of browsing the network or attempt to index all the available information in a central catalog. ALIBI (Adaptive Location of Internetworked Bases of Information) is a new tool that succeeds in locating information without the use of centralized resource catalogs, navigation, or costly searching. Its powerful query-based interface eliminates the need for the user to connect to one network site after another to find information or to wrestle with overloaded centralized catalogs and archives. This functionality was made possible by an assortment of significant new algorithms and techniques, including classification-based query routing, fully distributed cooperative caching, and a query language that combines the practicality of Boolean logic with the expressive power of text retrieval. The resulting information system is capable of providing fully automatic resource discovery and retrieval access to a limitless variety of information bases.

Keywords: Caching, replication, resource discovery, retrieval, routing.

[1]Work done at University of Maryland Baltimore County, Baltimore, MD 21228 U.S.A. Author's present address is Bldg. 225 Rm. A266, NIST, Gaithersburg, MD 20899 U.S.A. Phone: 301-975-3266; Fax: 301-948-6213; E-mail: dave@case50.ncsl.nist.gov.

1. Introduction

The Internet has been one of the most important contributions to the modern computing environment. It has had a considerable impact on the way that many individuals and companies conduct their daily business. Although it has remained fairly stable despite an exponential rate of growth, some applications have suffered more than others. One application that has suffered quite a lot is the use of the Internet as a source of information – what some would call a digital library.

When the number of public archives on the net became large, the resource discovery problem arose[Schwartz, 1988]. Users found themselves navigating file system after file system with FTP in search of specific files and having diminishing success at finding them. Riding to the rescue was Archie[Deutsch, 1992b, Obraczka et al., 1993], a centralized catalog that simply compiled the names of files in the directory trees of participating archives. Users who knew the name of the file they wanted could search Archie to find an accessible copy, then use FTP to retrieve it.

As the Internet continued to grow, Archie began to bog down. The volume of data that needed to be catalogued and the volume of queries over that data made rapid response impossible during peak usage, even with replication of the Archie database. However, the more significant problem was with the database itself. It contained only filenames. Although Archie helped users to find files once they knew the names of those files, it did not support any kind of content-based searching. With the huge volume of data available on the Internet constantly growing, it became more and more likely that any given piece of information was out there *somewhere,* but we lacked the ability to find it.

A complete solution to this problem has not yet appeared in the Archie system. A *whatis* database was added that permitted centralized indexing of software with brief descriptions, but it does not seem to have reached critical mass. At any rate, given the tendency of the filename database to bog down, we can easily imagine how much worse the situation would be if every file in the filename database also had an entry in the *whatis* database.

Centralized indexing cannot be the answer to the resource discovery problem in the Internet. Although its growth will eventually slow from an exponential to a geometrical rate once the number of Internet addresses per capita reaches the limit of convenience, it is already too big for centralized indexing and it is still growing exponentially. Decentralized tools like WWW[Berners-Lee et al., 1992, Andreessen, 1993], Gopher[McCahill, 1992] and WAIS[Obraczka et al., 1993, Kahle, 1991] resulted in part from our experiences with Archie. Unfortunately, in trying to solve some of the fundamental problems faced by decentralized systems, many of the new tools have re-introduced centralized indexing in some form or another. Each of them has been successful so far, but the indices are growing, and the costs are again mounting.

With this work, we offer a new alternative to centralized cataloging of resources, navigational resource discovery, direct manual file transfer, and massive centralized archives. Our alternative is named ALIBI, for Adaptive Location of Internetworked Bases of Information. To support networked resource discovery and information retrieval without resorting to any of the problematic techniques that have limited other systems, it was necessary for us to develop an entirely new approach. A combination of novel algorithms and careful design, developed in advance and improved along the way to the final implementation, made possible the software system we now have.

ALIBI consists of a network of information servers and a collection of information bases. The information bases are affiliated with individual information servers. The network of ALIBI, running as an application layer above the Internet, is called the Übernet. The servers are called Unetds (Übernet daemons). ALIBI offers all of the following services in a single system:

- Completely Automatic Resource Discovery

  The information servers accept queries from users (via a client program) and either arrange for them to be answered by the information bases affiliated with them or forward the queries to other servers that might be able to do so. Users provide queries that describe what they want, and the information system attempts to locate and fetch it. Users do not need to indicate the source of the information they desire, traverse a hierarchical file system, or navigate a hyperdocument to get to the information. Large, centralized archives and meta-indices are unnecessary because ALIBI locates the data wherever they lay without user assistance.

- Fully Distributed Resource Discovery

  ALIBI finds information without the use of any centralized services or algorithms. A special, fully distributed point-to-data routing algorithm is used by each site to forward queries towards the data they seek[Flater and Yesha, 1993]. Neither broadcasting of queries nor broadcasting of metadata are required to support the routing of queries. The information used to route queries is gleaned from resource providers and from response messages as they are passing through on their way to the sites that requested the information. Direct contact with distant sites is not needed.

- Cooperative Caching

  Some of the waste and redundancy present in file transfers today is eliminated by ALIBI's resource discovery mechanism, since it is likely to find the closest available source of data. Much of the remaining waste and redundancy is eliminated by ALIBI's integrated cooperative caching[Flater and Yesha, 1994]. Not only are retrieved data cached for reuse at the same site, they are cached cooperatively by groups of sites for the common good of all. The fully distributed caching mechanism uses cache space at underutilized sites to help out sites that are overburdened. The administrator at each site is free to choose how much cache space to provide; the system automatically adjusts to tolerate imbalances.

- Arbitrary Topology Information Network

  The list of sites hosting the information servers with which the local server should attempt to communicate must be provided by the local administrator. However, providing this list should not cause stress to anyone. It is not necessary to update the lists on both sides of a connection for the connection to be established; neither is it necessary to update the list whenever neighboring sites (those with which there is direct communication) appear or disappear. All this is sorted out automatically and transparently by the servers. Both the point-to-point and point-to-data routers are topology-independent. Although it is recommended that geographically nearby sites are chosen as neighbors in the information network, the administrator has complete freedom to choose whichever and however many sites he or she sees fit. Connections that offer poor performance are simply disused by the information system.

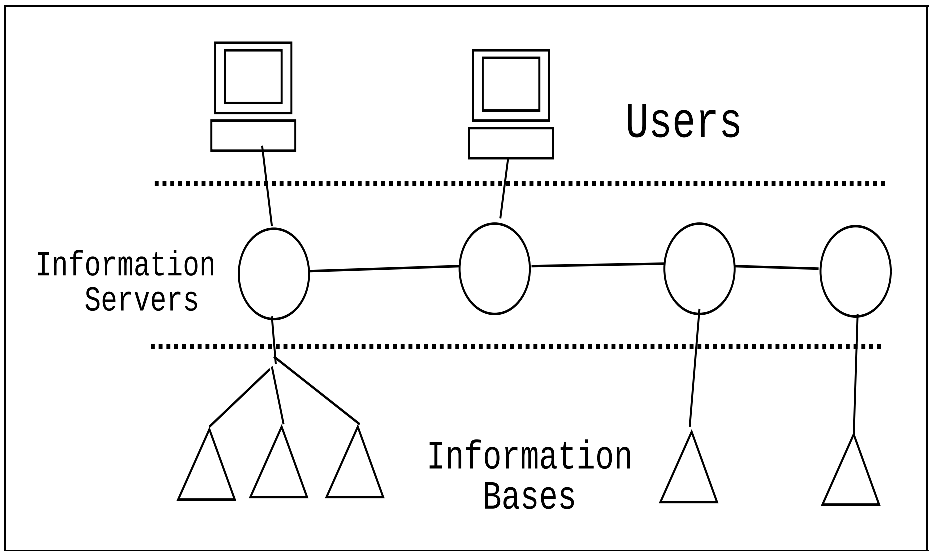
- Self-Maintaining Information Network

  The query router and other ALIBI components automatically adapt to shifting network topology. Broken connections are taken in stride, and periodically the effort is made to re-establish them. The abrupt appearance and disappearance of information servers does little to disrupt the normal operation of the information system as a whole. Unlike WWW/Mosaic and Gopher, ALIBI is not phased by a missing information base. It simply locates an alternate without bothering the user.

- Optimized Information Flow

  ALIBI routes response messages along the paths of least delay. The routing of query messages is patterned after the routing of response messages to the extent that the same low-cost paths are used. Since these paths conform to the characteristics of the underlying network and not to the unrelated topology of the information space, queries do not zigzag across long distances to locate data. Because of the flexibility of ALIBI's routing mechanisms, ALIBI is sometimes able to move data more reliably than direct FTP or telnet by avoiding unreliable chunks of the network.

Figure 1: ALIBI's Layered Information System Model



2. The ALIBI Information System

2.1. Resource Model

A key step in the development of ALIBI was determining a resource model. We needed to decide exactly what was the best way to partition the work between the general-purpose information servers and the domain-specific mediators[Wiederhold, 1992] that would access individual information bases. Our goal was to find an orthogonal, flexible model that would simplify the task of installing new mediators as more diverse information bases became available, but without placing unnecessary restrictions on the nature or complexity of the mediators.

The format of queries given to ALIBI was originally a list of keywords or simple English. A "junk words" list was used to eliminate irrelevant English verbiage from the query, a global thesaurus was used to replace query terms with standardized equivalents, and the query was then passed to the mediators to match as many keywords as possible using whatever mechanisms they chose to employ. As the system grew larger and we began to think about how to incorporate some non-fuzzy Boolean logic into the query language, it became clear that our approach was inadequate. Too much work was being done by the servers. Thesaural translation is a task that must be done within the specific domain of a particular information base, not at the global level. Furthermore, it is unsafe to remove "junk words" from a query in a global information system such as ALIBI. They might be acronyms for something important, or they could be significant in another language.

Since it was obvious that major changes would be needed, we scrapped our existing interface and rebuilt it based on a coherent resource model. This investment of time and effort quickly paid for itself with much easier development and installation of mediators.

Our resource model requires the following definitions:

(i) Mediators

A mediator is an entity that communicates with an information server to provide access to a single logical resource (information base). The mediator might actually coalesce several physical resources into a single logical resource, but this action is transparent to the information server. Each information server may be connected to any number of mediators, but there would be little advantage in having a mediator communicate with multiple information servers when those servers can communicate with one another.

(ii) Object Identifiers

An object identifier, or OID, uniquely identifies a single datum in the entire information system. The mediator for the information base containing that datum must be able to retrieve the datum when presented with its OID. For the purposes of the resource interface, the OID would only need to be unique within each resource, since it is not necessary for any external components to be able to locate the resource given only an OID. However, OIDs have other uses that do require global uniqueness.

(iii) Subqueries

A subquery is a list of keywords that can be processed by the mediator to yield the set of OIDs identifying data in its information base that "match" the subquery.

(iv) Matching

A datum matches a single keyword if it has a very high degree of relevance to that keyword according to the domain-specific logic used by the mediator. A datum matches a subquery if it has a very high degree of relevance to *each* of the keywords in the subquery. This insures that adding more terms to a subquery will make it more specific.

(v) Queries

A query is something that is processed by an information server, not by a mediator. Mediators only need to respond to subqueries. The query language understood by the information server can be arbitrarily complex.

In theory, one could simplify matters even more by defining subqueries to be single keywords and forcing the conjunctive logic to be performed by the information server. In practice, this would be extremely inefficient for large information bases, and any semantics that the information base might give to associations between several keywords could no longer be useful.

A resource is modeled as an entity that provides the following minimal set of services:

(i) Process a subquery

When presented with a subquery, the mediator must return a set of OIDs that identify matching data.

(ii) Fetch and classify a datum

When presented with an OID that it earlier returned in response to a subquery, the mediator must return the datum that the OID identifies along with an accurate classification of that datum.

(iii) Declaration of class (optional)

The resource may inform the information server of the class of data it contains. The information server can then save time by not processing queries against resources that contain the wrong kind of data.

This minimized resource model is powerful enough to support many non-trivial forms of information retrieval, including that of Alibi. The query language processed by Alibi information servers is described in the next section.

2.2. Processing and Routing of Queries

Our query language has the rare quality of combining fuzzy and non-fuzzy semantics in a coherent manner. It was designed to allow simple implementation, rather than user-friendliness, but it would not be a major task to have the client translate from a more intuitive format. The elements of the language are:

- Subqueries between parenthesis: `(cache software)`
- Individual OIDs in brackets: `[object-key host-id object-version]`
- Binary AND operator: `&`
- Binary OR operator: `|`
- Unary NOT operator: `~`

Expressions are built with postfix ordering[Kruse, 1987], so disambiguating parenthesis are not needed. The ability to specify particular OIDs in queries was included specifically to support the implementation of a **more** command in the client. If the datum identified by `[OID]` were returned in response to `(sound)(index)~&` and the user typed "more," the query `(sound)(index)~&[OID]~&` would automatically be submitted. Repeated use of the **more** command adds additional clauses to the query until finally there are no more data to be returned and the query fails. This automatic query construction gives the user a simple way to retrieve more data like the one just retrieved while preserving the simplicity of the query processor.

To process a full query, the information server sends the subqueries to the information base to be replaced with sets of OIDs, performs Boolean operations on these sets to produce a single set of OIDs, and, if the resulting set is not empty, instructs the information base to retrieve one of the data identified by an OID in the set. If there are multiple OIDs in the final set, one is chosen at random by the information server. The user is free to retrieve the others using the **more** command.

Accurate routing of queries is accomplished by sending queries towards sites that have answered similar queries in the past. The query classification algorithm, which determines this similarity, operates by examining the query for keywords that it recognizes. Data classes in ALIBI are specified by lists of keywords having decreasing significance, such as "blob software msdos." For this example, "blob" is the major class of data.[2] "Software" is a type of blob, and "msdos" is a type of software. Each word thus adds additional specificity to the more general class preceding it. Classes can be generalized by simply removing words from the end. Although a hierarchical class structure has its limitations, we were forced to make a decision when ALIBI was ready for implementation, and this is what we chose as the most practical approach.

"U" is defined to be the universal class, of which all others are a subclass. All classes except "U" can be considered to have an implied "U" at the beginning, and sufficient generalization of any class eventually results in "U."

2.3. Mediation

Each information base or resource is made available to ALIBI by a mediator connected to a single Unetd. Whatever operations are necessary to translate ALIBI subqueries into queries that are understood by the information base are done by the mediator. The protocol spoken by Unetd to its mediators is extremely simple, consisting of a RESET command, a FETCH command, and subqueries that are just lists of keywords. Mediators respond to subqueries and FETCHes with lists of object identifiers and raw data. They also have the ability to declare the specialty (class) of the provided resource to Unetd so that they will not be bothered with irrelevant queries.

In order to open a communication channel with the information server, the mediator creates and opens two named pipes in the directory used by Unetd. Unetd periodically scans its directory for named pipes. When it finds them, it opens them and immediately unlinks them. The pipes then disappear from the file system, but continue to exist as channels of communication through memory. The resource is then available to Unetd.

OIDs are unique in the entire information system because they are constructed with three fields: an object key provided by the information base, the network address of the site running the information server connected to the information base, and a version number or timestamp used to distinguish older and newer versions of the same data object. An example of an OID is

```
[/home/faculty/alibi/blobs/sound/00-index.txt -2108333000 754087762]
```

The first field, the object key, in this case is the full path name of the file being referenced. The resource that produced this OID is a generic "blobs" resource that simply treats entire files as individual data, so the full path name of the file makes a perfect object key. The next field is the Internet address of the machine running the server, daisy.cs.umbc.edu, output as an integer. The last field, the version number or timestamp, is the time at which the file was last modified according to the Unix **stat** function, expressed in seconds since 00:00:00 GMT 1/1/70 as is the Unix tradition. The format of the first and third fields will vary from one information base to the next, but the interpretation is the same: the first field is character data that identifies the data object, the last field is a number such that a higher number in the third field with the first and second fields being identical indicates a newer version of the same datum.

3. Example Resources

3.1. Cache Resource

The cache resource is always at the head of the resource list and the first to be checked for needed data. It is not an external process, but in most other respects it appears to the resource manager like any other resource. From the perspective of the resource manager, the only special thing about the cache is that a different function call is used to communicate with it. Although the fact that the cache has type "U" instead of something more specific is unusual for a resource, it is not inconsistent.

The function call provides one parameter to the cache resource that is not given to remote resources: the classification of the query. Most resources do not need this since they declare themselves to have a specific type and they are guaranteed not to be bothered with queries that are not even close. The cache, on the other hand, has no particular type but needs to match the query with replicas (cached responses) that are of the correct class. Most resources have the advantage of domain-specific knowledge to assist with matching queries to data; the cache has no such advantage.

When presented with a subquery, the cache returns the OIDs of all replicas with a matching class whose descriptions contain every keyword in the subquery, with the exception that keywords constituting part of the query's class are considered to have been matched already. It would be risky to use a fuzzy matching algorithm for the cache since it is general-purpose; it is not risky for other resources to use domain-specific thesauri to help with matching.

The class of a query is considered to match the class of a replica if the query class is the same as or a generalization of the class of the replica, or if either class is "U."

3.2. Blobs Resource

For our first real resource, we implemented a generic blobs resource that could be used to turn any collection of indexed files into an information base. We have used this mediator with an NFS connection to provide the MS-DOS subtree of wuarchive.wustl.edu, using the index files that were already there. The only customizations were to parse the slightly different format of wuarchive's index files and to generate resets when repeated NFS failures occurred. Isolated retrieval failures are assumed to mean that the index is out-of-date, and the relevant index entries are deleted. Repeated failures indicate an NFS outage, so the mediator informs Unetd that it is "down" until it succeeds in establishing a new connection and rebuilding its index.

Figure 2: A Session with ALIBI

```
Enter a query now, or enter ´quit´ to quit.
(msdos cache software)

Waiting for reply....
Response created at zing.ncsl.nist.gov Sun Apr 17 09:25:24 1994
Response path: zing.ncsl.nist.gov topdog.cs.umbc.edu greyhound.cs.umbc.edu
This datum has been in cache since Fri Apr  8 13:09:08 1994
Response created at retriever.cs.umbc.edu Fri Apr  8 13:08:40 1994
The following datum was provided by a blobs resource.
ID: /net/wuarchive.wustl.edu/archive/systems/ibmpc/msdos/diskutil/pckwk319.zip
 -2108333024 614736000
Class: blob software msdos diskutil
Description: pckwk319 zip  B   16358  890625  PC KWIK shareware disk cache
 program  v3 19 diskutil
Binary data follows ... 16358 bytes, filename pckwk319.zip

Warning:  existing files will be appended without further warning
Choose filename in which to save response [pckwk319.zip]:
Saving response....
Query path: greyhound.cs.umbc.edu zing.ncsl.nist.gov
Start time: 766589093
Answer for 185 at greyhound.cs.umbc.edu
Have a nice day =|-)

Enter a query now, or enter ´quit´ to quit.
more
(msdos cache software)[/net/wuarchive.wustl.edu/archive/systems/ibmpc
  /msdos/diskutil/pckwk319.zip -2108333024 614736000]~&

Waiting for reply....
Response created at retriever.cs.umbc.edu Sun Apr 17 09:25:37 1994
Response path: retriever.cs.umbc.edu greyhound.cs.umbc.edu
The following datum was provided by a blobs resource.
ID: /net/wuarchive.wustl.edu/archive/systems/ibmpc/msdos/diskutil/cacheart.zip
 -2108333024 588816000
Class: blob software msdos diskutil
Description: cacheart zip  B   13858  880829  Review of disk cache programs
 diskutil
Binary data follows ... 13858 bytes, filename cacheart.zip

Warning:  existing files will be appended without further warning
Choose filename in which to save response [cacheart.zip]:
Saving response....
Query path: greyhound.cs.umbc.edu topdog.cs.umbc.edu zing.ncsl.nist.gov
 sunset.ncsl.nist.gov retriever.cs.umbc.edu
Start time: 766589126
Answer for 185 at greyhound.cs.umbc.edu
Have a nice day =|-)

Enter a query now, or enter ´quit´ to quit.
quit
```

We are also using the blobs resource to provide a group of image files at NASA GSFC. This time, we needed to customize the mediator to add some additional lines to response messages that explain the copyrights on the files that were generated by employees of IBM.

Since the blobs resource is generic, non-fuzzy retrieval is the default. To generate internal descriptors against which to match query terms, the complete path and name of the file and any descriptions found in indices are concatenated and all punctuation marks are removed. When the blobs resource is applied within a domain, as we have applied it to MS-DOS software at wuarchive, the stock retrieval mechanism can be upgraded to a domain-specific fuzzy retrieval method.

The type that a blobs resource declares for itself ("blob software msdos" for wuarchive) is augmented by the directory names in the path to a file in order to form the classification for that file. The path is relative to whatever directory is specified as the "root blob directory." For wuarchive via NFS, /net/wuarchive.wustl.edu/archive/systems/ibmpc/msdos is the root blob directory. A file appearing in /net/wuarchive.wustl.edu/archive/systems/ibmpc/msdos/gnuish is therefore classified as "blob software msdos gnuish."

3.3. Usenet News Resource

The Usenet news resource turns newsgroups into information bases. Since we did not have direct access to the news spooling directory at any site, we implemented this resource using NNTP (Network News Transfer Protocol). All information is fetched remotely from an NNTP server. As ALIBI grows, we hope that we will be allowed to access one or more news spools directly to enhance performance.

Query keywords are matched against descriptions that contain the subject lines from news articles along with the names of the newsgroups to which those articles belong. Matches between subqueries and descriptions are made by the same algorithm used in the blobs resource. General queries match the description words derived from the names of newsgroups; specific queries match the description words derived from subject lines. The internal index of descriptions is periodically refreshed to keep up with the appearance of new articles and the expiration of old ones.

OIDs are formed by making the first field a combination of newsgroup and article number and making the version always zero. The version is not necessary because netnews assigns article numbers in a serial fashion, even when a new article supersedes an old one. Our decision to combine article number and newsgroup to form the ID instead of using the unique article IDs assigned by netnews was a matter of programming convenience and efficiency. It would not make much sense in ALIBI to have multiple sites providing the same newsgroups, so there is no harm in using this kind of OID.

Classifications for articles are formed by prepending "news" to the list of words in the name of the applicable newsgroup. For example, all articles in alt.politics.correct are classified as "news alt politics correct." We adopt the netnews hierarchy as a subtree of our classification system in the same way we adopted the directory structure of blob archives.

3.4. Software Reuse Library

As an example of how diverse kinds of retrieval can be supported by ALIBI, we modified the blobs resource to create a primitive software reuse library for C language source code. In the past, work has been done to support the retrieval of source code using variations on methods employed by full-text retrieval systems[Maarek et al., 1991]. To avoid the need to re-implement these complex algorithms for the sake of a single, prototype mediator, we settled for the simple heuristic of extracting the first nontrivial line of a `*.c` or `*.h` file to serve as a descriptor. Quite often, the comments at the head of a program begin with a one-line description of the program. For example, the beginning of the `ppmtogif.c` source file from the PBMPLUS distribution is shown in Figure 3.

The line "ppmtogif.c - read a portable pixmap and produce a GIF file" is read by the mediator and used to index `ppmtogif.c`. Although this heuristic only works when the programmer follows the convention of starting the program with a one-line description, it suffices for the purpose of demonstrating ALIBI's ability to support software reuse. It also serves as a trivial example of how automatic indexing can be used by ALIBI resources having specialized domains.

Figure 3: Header comments from ppmtogif.c

```
/* ppmtogif.c - read a portable pixmap and produce a GIF file
**
** Based on GIFENCOD by David Rowley <mgardi@watdscu.waterloo.edu>.A
** Lempel-Zim compression based on "compress".
**
** Copyright (C) 1989 by Jef Poskanzer.
**
** Permission to use, copy, modify, and distribute this software and its
** documentation for any purpose and without fee is hereby granted, provided
** that the above copyright notice appear in all copies and that both that
** copyright notice and this permission notice appear in supporting
** documentation.  This software is provided "as is" without express or
** implied warranty.
**
** The Graphics Interchange Format(c) is the Copyright property of
** CompuServe Incorporated.  GIF(sm) is a Service Mark property of
** CompuServe Incorporated.
*/
```

The classification of data and assignment of OIDs by the source code resource are identical to what is done by the blobs resource.

3.5. SEC EDGAR Resource

EDGAR is an FTP archive that is sometimes available to the public at town.hall.org. It contains the electronic versions of forms filed with the SEC (Securities and Exchange Commission) by companies in the U.S. The names of the files containing the forms are cryptic, but the files are indexed by company name in a separate index file. We modified the blobs resource to access the EDGAR database with anonymous FTP.

When it is reset, the Edgar mediator FTPs the index and reads it into memory. Subqueries are then matched against company names. Matching files are retrieved using FTP and given to the Unetd as blobs. Although the information is all textual, most of the forms are large enough that it is better to treat them as blobs.

All data fetched from the EDGAR database are classified as "government USA SEC EDGAR." OIDs consist of the relative pathname of the file, the host ID, and a null version number since individual forms are never altered.

3.6. Geographical Database

Henry Tom of NIST provided us with a geographical database conforming to FIPS 55-2. The format specified by FIPS 55-2 is a flat file containing 132-character records divided into fixed-length fields. The Edgar mediator was the basis for the FIPS 55-2 mediator since it had already been substantially simplified from the blobs mediator.

The FIPS 55-2 mediator matches subqueries against place names and returns individual records from the database, along with a brief explanation of what they mean. A FIPS 55-2 record contains a substantial amount of encoded information. The entire record is provided as part of the response, but only the most significant fields are explained by the mediator. An example is shown in Figure 4. The 132-character record has been broken up to fit on the page.

Responses are classified as "government USA census geography" followed by the post office abbreviation for the state to which the records pertain. At the time of this writing we only have the

Figure 4: Sample Response from FIPS 55-2 Geographical Database

```
Response created at sunset.ncsl.nist.gov Tue Mar  8 13:53:19 1994
Response path: sunset.ncsl.nist.gov cesdis7.gsfc.nasa.gov
 dunloggin.gsfc.nasa.gov greyhound.cs.umbc.edu retriever.cs.umbc.edu
The following datum was provided by the NIST FIPS 55-2 Geographical
 Database (Virginia)
Thanks go to Henry Tom for this database.
ID: 6911 -2130298111 0
Class: government USA census geography VA
Description: Wolf Trap Farm Park For The Performing Arts
FIPS 55-2 Record:  5187256VA 1 1M4Wolf Trap Farm Park For The
   Performing Arts  059Fairfax           22180           8840 1110
A brief summary of the significant features of this record follows.
-     The place is called Wolf Trap Farm Park For The Performing Arts
-     It's a federal facility
-     It is in the state of VA
-     Its county is Fairfax
Query path: retriever.cs.umbc.edu cesdis7.gsfc.nasa.gov
 sunset.ncsl.nist.gov
Start time: 763152827
Answer for 16 at retriever.cs.umbc.edu
Have a nice day =|-)
```

data for Virginia on line, but this will probably change. OIDs are constructed by assigning a unique serial number and a null version number to each record.

4. Alibi vs. Other Approaches

Alibi is not the only tool capable of handling the diverse types of information that we have described, but it is the only one to support fully automatic resource discovery. In the following subsections we will compare Alibi with each of the currently popular alternatives.

4.1. Archie/FTP

FTP (File Transfer Protocol)[Postel and Reynolds, 1985] has been the primary method of information transfer on the Internet since the net reached critical mass. Anonymous FTP, which eliminates the need to have an account on the remote machine to retrieve data from it, has been the primary information retrieval mechanism for public data. The FTP commands useful for information retrieval are **dir** or **ls** (list files in this directory), **cd** (change to another directory), and **get** (fetch a file).

Since FTP alone provides little assistance for resource discovery, Archie was built. Today it remains the best tool for locating data available via FTP, but it is overburdened. When centralized indexing is used to solve the resource discovery problem in a growing, decentralized system, the Archie syndrome results. The index becomes an unavoidable bottleneck. The Archie syndrome keeps being repeated because the problems with centralized indexing only become apparent *after* the system reaches critical mass, and the exponential growth of the network is transformed into exponential index growth.

The Archie/FTP combination has other problems. For example, FTP is very inefficient in its use of network resources. Nearly 40% of FTP transmissions are redundant[Ewing et al., 1992], yet there is no caching. This problem is solved by some of the extended networked file systems we will describe later, but resource discovery remains a problem for them.

4.2. WWW/Mosaic

The World Wide Web[Obraczka et al., 1993, Berners-Lee et al., 1992] is the latest and greatest of a series of networked tools following in the footsteps of Gopher and WAIS (described below). The Web appears to the user as an enormous hyperdocument. Resource discovery is achieved almost entirely through navigation, but full-text retrieval is supported by some WWW servers once you find them. WWW interfaces with WAIS, Gopher, netnews, and FTP.

WWW is complemented by an X Windows based client program called Mosaic[Andreessen, 1993]. Many other WWW clients exist, but Mosaic has become exceedingly popular. Mosaic completes the hypertext approach enabled by WWW with a graphical user interface. Hypertext links can be traversed by clicking on highlighted items in the text. When links point to data at other servers, Mosaic connects to the other server and retrieves the referenced data.

WWW and Gopher both suffer from stale metadata, i.e. links pointing to data that no longer exist. It often happens that links are not redirected or deleted when data are relocated or removed. An attempt to traverse such a link causes an error message to be returned to the user. Some consider this to be a significant problem; others simply shrug it off and continue to navigate. The analogous situation in ALIBI is when a query is misrouted because the information in some query routing table is stale. When this happens, ALIBI silently recovers and tries alternate routes until the data are found. The routes of ALIBI queries are not "hard coded" like links in WWW and Gopher; the query routing tables adapt to changes in the location of data and in the topology of the information network. The possibility that there will be stale metadata at the mediator level in ALIBI is handled with the OOPS protocol unit. If a mediator returns OIDs to Unetd that reference unretrievable data and Unetd then tries to FETCH one of them, the mediator can return OOPS. This event is just as transparent to the user as the recovery from misrouted queries.

We do not intend to argue that stale metadata are a reason to choose ALIBI over WWW. We believe that ALIBI's fully automatic resource discovery and query-based interface are a valuable alternative to resource discovery that requires navigation. Instead of navigating to the correct site and retrieving data, or searching a catalog for the correct site and retrieving data, ALIBI users simply retrieve the data. This results in less work for the user and eliminates the network traffic generated by navigation to faraway sites. Retrieval of data from distant sites is also reduced by ALIBI since servers between them and the client (not just the endpoints) have a chance to cache responses.

4.3. Wais

WAIS[Obraczka et al., 1993, Kahle, 1991] (Wide Area Information Servers) began as a full-text retrieval system, and that remains its primary application. The user must select a group of servers against which to execute searches, then submit a keywords query to perform the search. Modern text retrieval algorithms are used to estimate the relevance of documents to a query, and the user is given a list of relevant documents that may be retrieved.

Since searching every WAIS database is not practical, the user must determine in advance which servers should be included in a search. To help with this, a centralized meta-index is maintained that lets users retrieve the names of relevant servers before actually executing the query.

The primary contribution of WAIS was a standard for access to remote resources. WAIS was among the first in the genre of information systems to which Gopher, WWW, and ALIBI also belong; they all attempt to make the world available through a single user interface. Its primary limitation is that it has the Archie syndrome. The user must consult a centralized index before retrieving data, or else manually select databases from a list.

4.4. Gopher

Gopher[Obraczka et al., 1993, McCahill, 1992, Lindner, 1994] is primarily a hierarchical, navigational information system. It is not completely hierarchical because a "child" node may actually lead back to a parent; it is not completely navigational because individual Gopher servers can support full-text searches. To the user, Gopherspace appears as a never-ending series of menus that lead from one to the other, with the occasional appearance of menu options that allow keyword searches at the Gopher server that is currently active.

Whether because of a limitation in Gopher or because of poor configuration by Gopher administrators, the emergent structure of Gopherspace never achieved a very high level of semantic linkage between different Gopher servers. This resulted in the Archie syndrome yet again. The meta-index for Gopher appears to be a tool called Veronica that indexes directory names and documents in Gopherspace and permits Archie-like searches. Gopher also offers access to WAIS, Archie, and FTP.

Although the Archie syndrome threatens Gopher, its biggest problem is that it is rapidly being supplanted by WWW. If the semantic links in Gopherspace were improved, Gopherspace might begin to resemble the Web. We believe it is not so much the added functionality of WWW, but rather its flashy user interface (Mosaic), that has lured users away from Gopher.

4.5. Extended Networked File Systems: Alex, WWFS, Prospero, ...

NFS[Sun Microsystems, 1989] (Network File System) and AFS[Deutsch, 1992a, Satyanarayanan, 1990] (Andrew File System) are widely used protocols that allow remote file systems to appear to be mounted onto a local file system. AFS supports caching to provide better performance than NFS, but otherwise they are very similar from the user's perspective. The success of NFS and AFS in facilitating remote file access has led researchers to expand on them to try to solve more of the Internet's problems. Alex[Obraczka et al., 1993, Cate, 1992] and WWFS[Kadobayashi,] are two software packages that translate NFS protocol requests into FTP requests. By installing these packages, it is possible to access much more information through NFS. One might be tempted to describe them as mediators for NFS; however, it would be more accurate to call them translators[Barbará and Clifton, 1992]. The NFS and FTP protocols both access hierarchical file systems. The operations supported are similar; only the protocol needs to be translated. The translator is making it possible to access more data, but not more kinds of data. Even if mediators were written to provide access to diverse data via NFS, the fact remains that NFS provides a hierarchical view of data with little support for resource discovery. NFS is excellent at what it does, but it is not a resource discovery tool.

Using a hierarchical distributed file system is just a special case of navigation in which the links that can be traversed form a tree. This restriction makes it more difficult for users to locate desired data. As the hierarchy grows without bound and navigation becomes increasingly frustrating, we soon find ourselves re-inventing Archie to help users find what they need, and we have the Archie syndrome yet again. The improvements of extended file systems as a class over Archie/FTP are file and metadata caching and the possibility of supporting attribute searches at a lower level than Archie[Obraczka et al., 1993].

The Prospero Virtual File System[Obraczka et al., 1993, Deutsch, 1992a, Neuman, 1992b, Neuman, 1992a] integrates NFS and FTP like the others but also supports the creation of virtual directories and interfacing with Archie, Gopher and WAIS. Virtual directories effectively contain links that can be traversed to access the listed data. Prospero has many similarities to Gopher, but has close ties with Archie. Suffice to say that all its features appear as features of other systems that we have already discussed, so without intending any slight of the Prospero system, we will not repeat ourselves.

4.6. Discussion

There have been many taxonomies of Internet resource discovery tools[Obraczka et al., 1993, Schwartz et al., 1992]; we have done another in light of the advances made by ALIBI.

Table 1 shows the result of our high-level analysis. Our categories for each tool are the resource discovery mechanism it uses to find appropriate sites, the retrieval mechanism used to select and fetch individual data once a site is found, the kind of optimization used to enhance performance, the method used to determine the metadata for resource discovery, and the inherent bias of the system in terms of what kind of data it most easily handles.

Table 1: Yet Another Taxonomy of Internet Tools

Tool	Discovery	Retrieval	Optimization	Metadata	Bias
Archie	AS	N/A (FTP)	Replication	Refresh	Filenames
WAIS	AS	FT	Replication	N/A	Text
Gopher, Veronica	Navigation, AS	Navigation, FT	Caching, Replication	Static	Hierarchical
WWW	Navigation	Navigation, FT	Caching	Static	Hypertext
ALIBI	Automatic	Flexible	Distrib. Caching	Dynamic	Atomic Data

AS = Archie Syndrome (search central index for relevant sites)
FT = Full-Text retrieval (inverted index keyword search)

In the first column, we see that ALIBI is the only system with automatic resource discovery. Archie makes it possible to search a filename index (or the *whatis* database) to try to locate something; as the prototype of all centralized catalog approaches to resource discovery, Archie defines the Archie syndrome. WAIS incurs the Archie syndrome at the server selection stage; to find out which servers should be searched for some data, the user must either browse through a long index of servers or execute queries on a centralized catalog. Gopher supports resource discovery through navigation (the equivalent of "manual" resource discovery); Veronica adds Archie-like searches and the Archie syndrome to Gopher. The improved semantic linkage of the Web has reduced the need for centralized indices, but several of them are already available a link or two away from the NCSA home page, and Web users are taking the initiative to create more.

ALIBI also offers a new and flexible retrieval method. Archie does not compete in this arena since it is not a retrieval tool. WAIS is primarily a full-text retrieval system. Gopher and WWW support full-text retrieval at specific servers once the resource discovery step is complete; however, they also provide the ability to locate specific data through more navigation. While ALIBI does not offer navigation, its two-tiered query language supports more flexible retrieval than full-text alone. ALIBI subqueries are interpreted by resource providers in a domain-specific manner, allowing many alternatives besides traditional inverted index searching over a text-only database.

Moving on to the optimization column, both WWW and Gopher permit caching to be done at a local server or by a client program. However, this caching is not cooperative in the way that ALIBI's caching is. In the Übernet, if one site generates many queries while a nearby site is idle, that site will accumulate cached replicas of responses for the busy site. Queries from other sites that are routed through these sites may also be answered from their caches. Gopher and WWW establish direct Internet connections to archive sites, making cooperative caching impossible and causing well-known archives to become overloaded. At the time of this writing, the "home page" for WWW at NCSA begins with "[The] NCSA Web server is overloading due to an exponential growth in connections." Replication is also used to optimize the performance of Internet tools. The centralized indices of Archie, WAIS, and Veronica have been replicated to keep up with the growing user load. ALIBI's caching effectively performs dynamic replication[Wolfson and Jajodia, 1992], creating more replicas of the data that are most in demand.

The metadata used for resource discovery by the various Internet tools are maintained in different ways, to different degrees. Archie updates its metadata by periodically refreshing it. Although it is costly, this keeps the metadata from becoming stale most of the time. The only WAIS metadata used for resource discovery is the index of WAIS servers, which is maintained by a central registry. The directory services embedded in Gopher and WWW presumably are refreshed like Archie, but most of the other links used for navigation are static, persisting indefinitely after the referenced data is removed and causing annoyance to users. ALIBI remains the exception, silently recovering from stale metadata in query routing tables without bothering the user and collecting metadata without incurring the cost of periodic refresh.

Finally, there is the issue of bias. Each information system has a bias for what kind of data it most conveniently handles. It is not necessarily the case that they *cannot* handle other types of data, but it is better to be biased towards more general data types than more specific ones. Archie primarily indexes filenames. WAIS is heavily biased towards full-text retrieval. Gopherspace consists almost entirely of self-contained hierarchies at different sites with little non-hierarchical linkage, so we give it a hierarchical bias. The Web has more non-hierarchical linkage and has multimedia support with Mosaic, so we say that it has a hypertext bias. The bias of ALIBI is for atomic data (self-contained data objects). For ALIBI to derive data from fragments at many sites, a mediator must coalesce the fragments and present them to ALIBI as a single object. However, the content of the data objects and the internal organization of each information base are arbitrary.

5. Conclusions and Future Work

ALIBI provides a completely new approach to networked resource discovery and information retrieval. In this paper we have given a brief overview of ALIBI's functionality and shown how ALIBI differs from the other systems. Those readers who would like a more thorough and technical discussion of ALIBI and its components can get a copy of the dissertation "Towards a Global Federation of Heterogeneous Resources" from the University of Maryland Graduate School, Baltimore. Software for the client and server are available for anonymous FTP from speckle.ncsl.nist.gov in flater/sources (Alibi1.0.tgz).

Future work will include producing a more complex client, further optimization of the distributed caching and classification algorithms, special handling of extremely large data objects, additional mediator development, and an e-mail interface.

Notes

1 Blob is an acronym for "Binary Large OBject." The creator of this acronym is unknown to us, but it is used by some SQL databases that have multimedia support.

References

[Andreessen, 1993] Andreessen, Marc (1993). NCSA Mosaic technical summary [on-line]. Technical report, Software Development Group, National Center for Supercomputing Applications, 605 E. Springfield, Champaign IL 61820. Available FTP: Hostname: ftp.ncsa.uiuc.edu Path: /Mosaic-/mosaic-papers File: mosaic.ps.Z.

[Barbará and Clifton, 1992] Barbará, Daniel and Clifton, Chris (1992). Information brokers: Sharing knowledge in a heterogeneous distributed system. Technical Report MITL-TR-31-92, Matsushita Information Technology Laboratory, 182 Nassau Street, Princeton, NJ 08542.

[Berners-Lee et al., 1992] Berners-Lee, T. J., Cailliau, R., Groff, J-F, and Pollermann, B. (1992). World-Wide Web: The information universe. *Electronic Networking: Research, Applications, and Policy*, 2(1):52–58.

[Cate, 1992] Cate, Vincent (1992). Alex – a global file system. In *Proceedings of the Usenix File System Workshop (Ann Arbor, Michigan)*, pages 1–11, Berkeley, California. USENIX Association.

[Deutsch, 1992a] Deutsch, Peter (1992a). Resource discovery in an internet environment. Master's thesis, School of Computer Science, McGill University.

[Deutsch, 1992b] Deutsch, Peter (1992b). Resource discovery in an internet environment–the Archie approach. *Electronic Networking*, 2(1):45–51.

[Ewing et al., 1992] Ewing, David J., Hall, Richard S., and Schwartz, Michael F. (1992). A measurement study of internet file transfer traffic. Technical Report CU-CS-571-92, Department of Computer Science, University of Colorado.

[Flater and Yesha, 1993] Flater, David W. and Yesha, Yelena (1993). An efficient management of read-only data in a distributed information system. *International Journal of Intelligent and Cooperative Information Systems*, 2(3):319–334.

[Flater and Yesha, 1994] Flater, David W. and Yesha, Yelena (1994). Managing read-only data on arbitrary networks with fully distributed caching. *International Journal of Intelligent and Cooperative Information Systems*, 3(3):279–292.

[Kadobayashi,] Kadobayashi, Youki. Department of Information and Computer Sciences, Osaka University, Japan. Articles posted to comp.sources.misc.

[Kahle, 1991] Kahle, Brewster (1991). Available FTP: Hostname: think.com Path: /public/wais File: README.

[Kruse, 1987] Kruse, Robert L. (1987). *Data Structures and Program Design.* Prentice-Hall, second edition.

[Lindner, 1994] Lindner, Paul (1994). Internet Gopher user's guide. Available FTP: Hostname: boombox.micro.umn.edu Path: /pub/gopher/docs File: GopherGuide_Jan12-94.ps.

[Maarek et al., 1991] Maarek, Yoëlle S., Berry, Daniel M., and Kaiser, Gail E. (1991). An information retrieval approach for automatically constructing software libraries. *IEEE Transactions on Software Engineering*, 17(8):800–813.

[McCahill, 1992] McCahill, M. (1992). The internet gopher protocol. In *Proceedings of the 23rd Internet Engineering Task Force*. CNRI.

[Neuman, 1992a] Neuman, B. Clifford (1992a). Prospero: A tool for organizing Internet resources. *Electronic Networking*, 2(1).

[Neuman, 1992b] Neuman, B. Clifford (1992b). The Prospero file system: A global file system based on the virtual system model. *Computing Systems*, 5(4):407–432.

[Obraczka et al., 1993] Obraczka, Katia, Danzig, Peter B., and Li, Shih-Hao (1993). Internet resource discovery services. *IEEE Computer*, 26(9):8–22.

[Postel and Reynolds, 1985] Postel, J. and Reynolds, J. (1985). File Transfer Protocol. RFC 959.

[Satyanarayanan, 1990] Satyanarayanan, Mahadev (1990). Scalable, secure, and highly available distributed file access. *IEEE Computer*, 23(5).

[Schwartz, 1988] Schwartz, Michael F. (1988). The networked resource discovery project: Goals, design, and research efforts. Technical Report CU-CS-387-88, University of Colorado, Boulder, Colorado 80309.

[Schwartz et al., 1992] Schwartz, Michael F., Emtage, Alan, and Kahle, Brewster (1992). A comparison of internet resource discovery approaches. *Computing Systems*, 5(4):461–493.

[Sun Microsystems, 1989] Sun Microsystems (1989). NFS: Network File System protocol specification. RFC 1094.

[Wiederhold, 1992] Wiederhold, Gio (1992). Intelligent integration of diverse information. In *Proceedings of the ISMM First International Conference on Information and Knowledge Management*, pages 1–7, Baltimore, MD, U.S.A. The International Society for Mini and Microcomputers.

[Wolfson and Jajodia, 1992] Wolfson, Ouri and Jajodia, Sushil (1992). Distributed algorithms for dynamic replication of data. In *Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data (San Diego, California)*, New York, New York. Association for Computing Machinery.

About the Authors

David Flater is currently a Computer Scientist in the Information Management Group of the NIST Information Technology Laboratory. Work for this paper was done at the University of Maryland Baltimore County. Correspondence should be sent to David Flater, Bldg. 225 Rm. A266, National Institute of Standards and Technology, Gaithersburg, MD 20899 U.S.A., or e-mailed to dave@case50.ncsl.nist.gov.

Yelena Yesha is a guest researcher at NIST and a professor at the University of Maryland Baltimore County. Correspondence should be sent to Yelena Yesha, Computer Science Department, University of Maryland Baltimore County, Baltimore, MD 21228 U.S.A., or e-mailed to yeyesha@cs.umbc.edu.